ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №1

«алгоритми обробки масивів»

з курсу «Алгоритми і структури даних»

студента групи ПА-18-2

Прозора Дениса Євгеновича

кафедра комп’ютерних технологій, ДНУ

2018/2019 н.р.

**1. Постановка задачі**

Розробити програму, яка реалізує такі дії:

- створює статичний масив із 100 елементів;

- заповнює масив *n* випадковими цілими числами із діапазону, границі якого ввести з клавіатури. Значення *n* (*n*<100) також ввести з клавіатури;

- виводить заповнену частину (із *n* елементів) на екран;

- виконує обробку масиву із *n* елементів згідно з завданням за варіантами

1. Знайти середнє арефметичне всіх парних елементів масиву. Результати показати на екрані.
2. Додати після кожного **від’ємного елемента його модуль. Вивести на екран масив до та після перетворення.**

**2. Опис розв’язку**

Для початку ми створюємо масив а[] на 100 елементів. Потім пропонуємо користувачу обрати розмір, який буду заповнено випадковими числами(межі яких мі також запитуємо у користувача та записуємо у n1 та n2).

Для перевірки ввода, використовується функція **check,** яка на вхід приймає 2 межі, та якщо ввід пройшов успішно то повертає це число, якщо ні, то користувач вводить число заново. Ще ми перевіряємо правильність ввода меж та якщо вони введені неправильно – пояторюємо процедуру ввода.

Далі ми заповнюємо наш масив випадковими числами відповідно межам за допомогою функції **random** та циклу. Потім використовуємо функцию **array\_cout** для виводу нашого масиву (на вхід функції подається масив та кількість його елемнтів і за допомогою цикла він їх виводить).

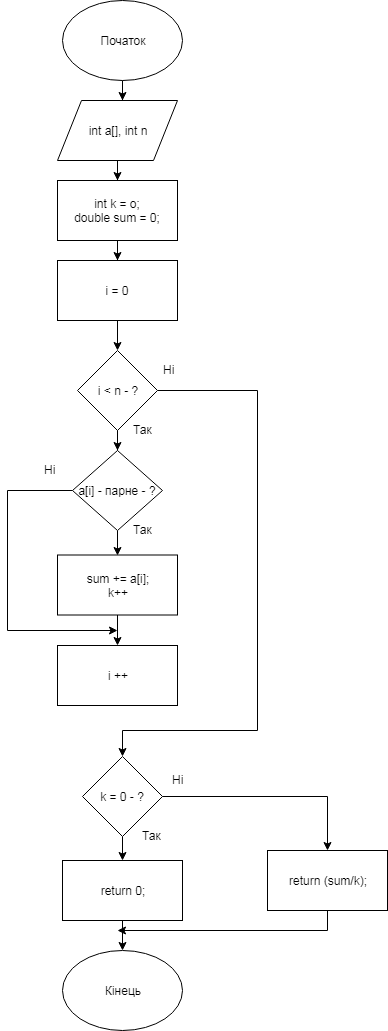
Далі запускаємо меню з вибором операції :

1. Виконаня першого завдання та повертання до меню.
2. Виконаня другого завдання та повертання до меню.
3. Вихід з меню.

Реалізуємо це чере **switch case,** у першому та другому випадку визиваючи функції відповідні до завдань. А у третьому просто повертаємо значення функції **main,** тим самим завершуючи програму.

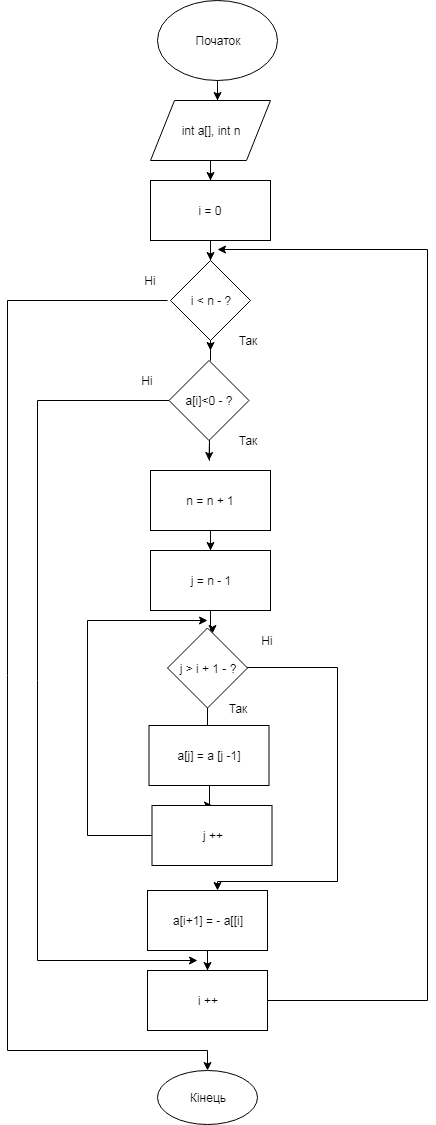
**Завдання 1 :**

Функція **average** приймає масив та його розмір, повертає **double.** Заводемо змінну **sum** для накопичення суми парних елементів масиву, **k** для накопичення кількость парних елементів. Запускаємо цикл де робимо перевірку, якщо число парне, то записуємо його до **sum** та збільшуємо лічильник **k**. Якшо у кінці **k** дорівнює 0, то повертаємо 0, якщо іншому то ділемо суму на кількість і повертаємо результат.



**Завдання 2 :**

Використаємо функцію **module\_check(**для перевірки чи не вийдемо за межі масиву), яка приймає масив та кількість його елементів та повертає true або falsе. Потім робимо цикл де в умові визиваємо функцію **module\_check.** Якщо все правильно, виведемо масив до операції та визовемо функцію **module** яка і виконає перетворення, після якого залишиться тільки ще раз вивести масив. Функція **module** приймає масив та кількість його елементів. Для виконання завдання ми використуємо 2 цикла, у першому шукаємо **від’ємний елемент(через цикл). Якщо знайшли, то збільшуємо кількість елементів на 1(n) та заходемо у другий цикл, де зміщуємо елементи масиву до нашого від’ємного(поточне і). Після цього в елемент після нашого від’ємного(поточне і) записуємо його модуль. Далі виходемо у перший цикл та прогоняємо так до кінця. У кінці ми отримуємо змінений масив, де після кожного від’ємного елемента його модуль.**



**3. Вихідний текст програми**

#include <iostream>

#include <ctime>

#include <stdlib.h>

using namespace std;

int check(int n1, int n2)

{

int n;

while(!(cin >> n) || (n > n2) || (n < n1))

{

cout << "Введено неверное значение. Пожалуйста, введите ещё раз : ";

cin.clear();

while (cin.get() != '\n');

}

return n;

}

void array\_cout(int a[], int n)

{

for (int i = 0; i < n; i++)

{

cout << a[i] << " ";

}

cout <<endl;

}

double average(int a[],int n)

{

double sum = 0;

int k = 0;

for(int i = 0; i<n; i++)

{

if(a[i]%2==0)

{

sum += a[i];

k++;

}

}

if(k==0)

return 0;

return (sum/k);

}

bool module\_check(int a[], int n)

{

int k =0;

for(int i = 0; i<n; i++)

{

if(a[i]<0)

k++;

}

if((n+k)<=100)

return true;

else

{

cout<<"Операция невозможна, так как выходит за границы массива";

return false;

}

}

void module(int a[], int &n)

{

for(int i = 0; i<n; i++)

{

if (a[i]<0)

{

n = n + 1;

for (int j = n - 1; j > i+1; j--)

{

a[j] = a[j-1];

}

a[i+1] = -(a[i]);

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

int a [100];

int n,n1,n2,k;

cout<<"Ведите размер массива(от 1 до 100) : ";

n = check(1,100);

cout<<"Ведите нижнюю границу значений массива: ";

n1 = check(-2000000000,2000000000);

cout<<"Ведите верхнюю границу значений массива: ";

n2 = check(-2000000000,2000000000);

while (n2 < n1)

{

cout << "Верхняя граница должна быть больше нижней." << endl;

cout<<"Ведите нижнюю границу значений массива : ";

n1 = check(-2000000000,2000000000);

cout<<"Ведите верхнюю границу значений массива : ";

n2 = check(-2000000000,2000000000);

}

for(int i = 0; i<n; i++)

{

a[i] = n1 + rand()%(n2-n1+1);

}

cout<<"Ваш массив : "<<endl;

array\_cout(a,n);

while(true)

{

cout << "Выберите операцию : " << endl;

cout << "1. Найти средне арифметическое всех четных элементов массива. Результат показать на экран." << endl;

cout << "2. Добавить после каждого отрицательного элемента его модуль. Вывести на экран массив до и после преобразования."<< endl;

cout << "3. Выход из программы." << endl;

k = check(1,3);

switch (k)

{

case 1 :

{

cout << endl;

cout<<average(a,n)<<endl;

cout << endl;

}

break;

case 2 :

{

cout << endl;

if(module\_check(a,n))

{

cout<<"До преобразования : "<<endl;

array\_cout(a,n);

module(a,n);

cout << endl;

cout<<"После : "<<endl;

array\_cout(a,n);

cout << endl;

}

}

break;

case 3 :

{

cin.get();

return 0;

}

break;

default :

{

break;

}

break;

}

}

return 0;

}

**4. Опис інтерфейсу**

На початку користувач вводить розмір масиву та його числові межі. Потім він бачить меню, де йому пропонуються дії, та вибирає яку операцію він хоче виконати(через ввод 1,2 або 3).

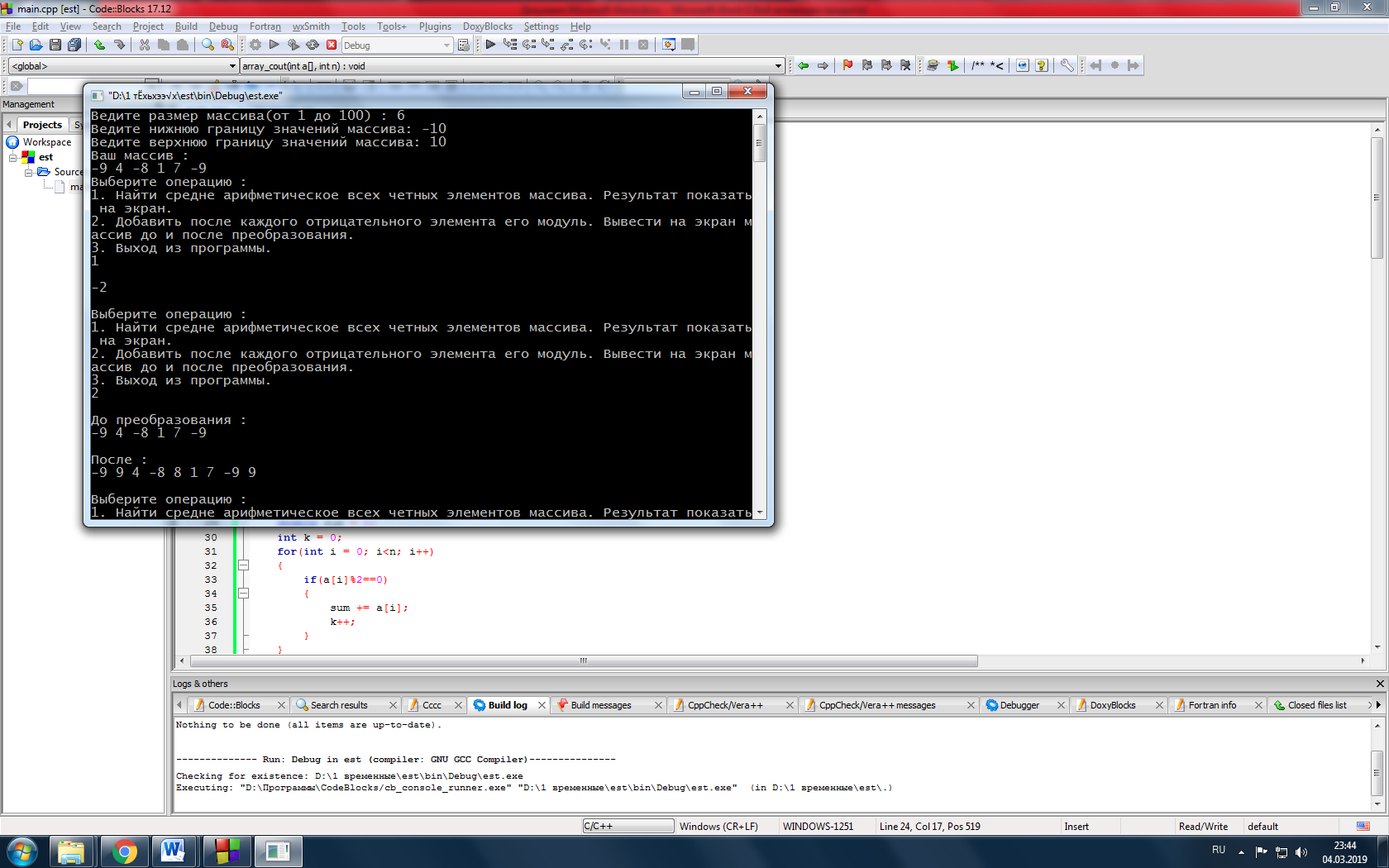
1 – виконання першого завдання.

2 – виконання другого завдання.

3 – вихід з меню.

Після кожного завдання у користувача є можливість вибрати ще.

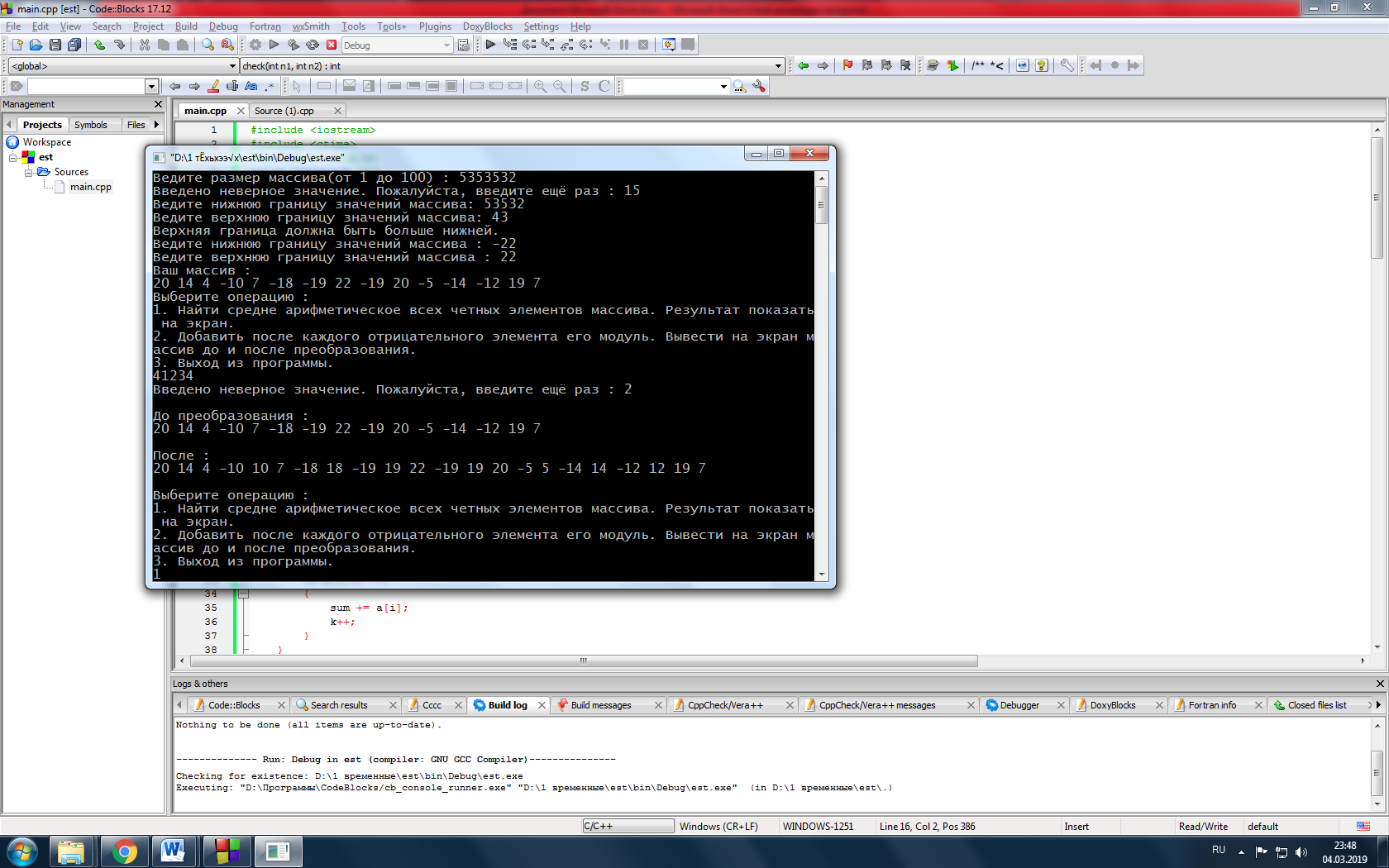
**5. Опис тестових прикладів та скриншоти їх реалізації.**



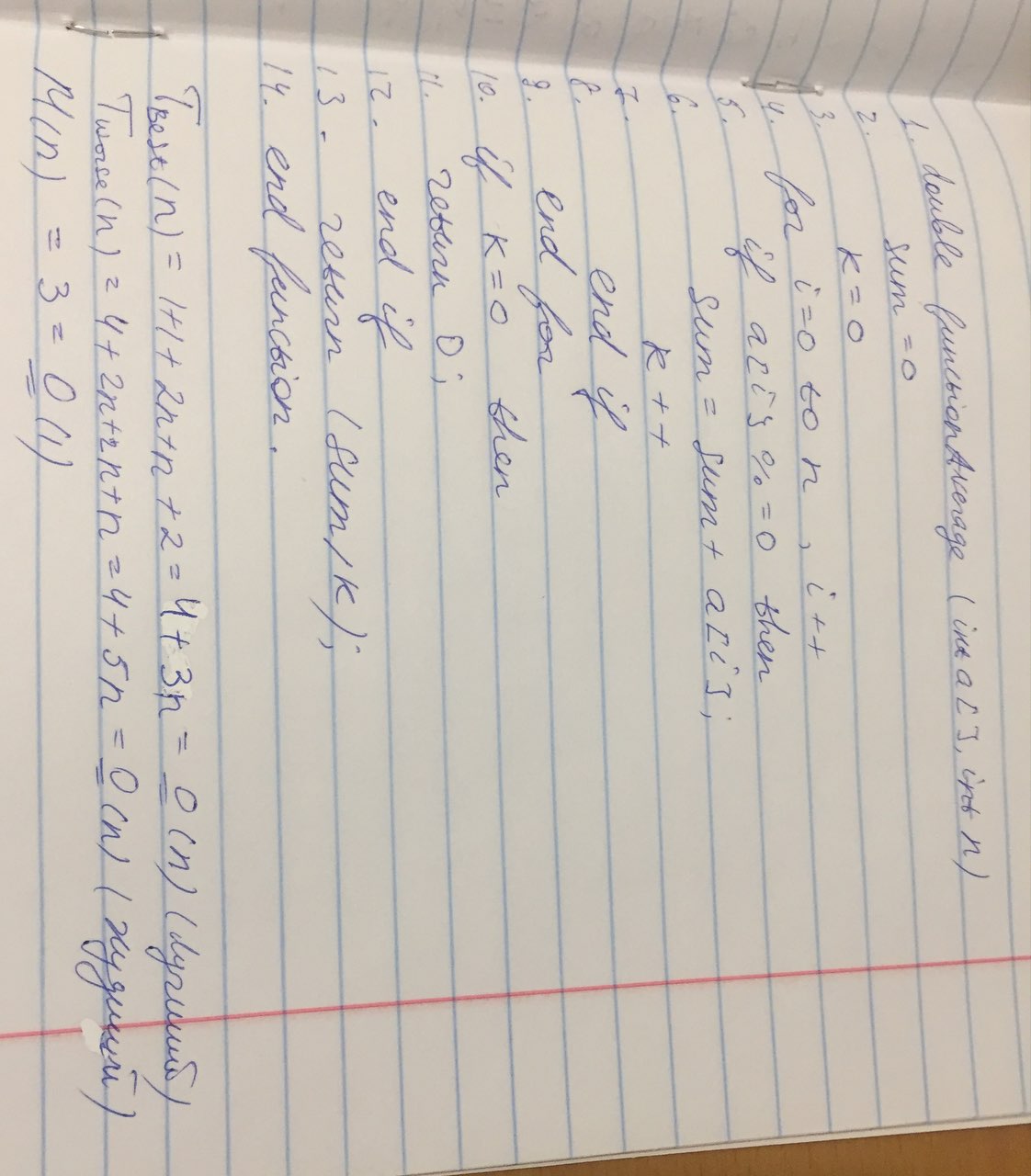
Перевіримо виконання операцій для n = 6.

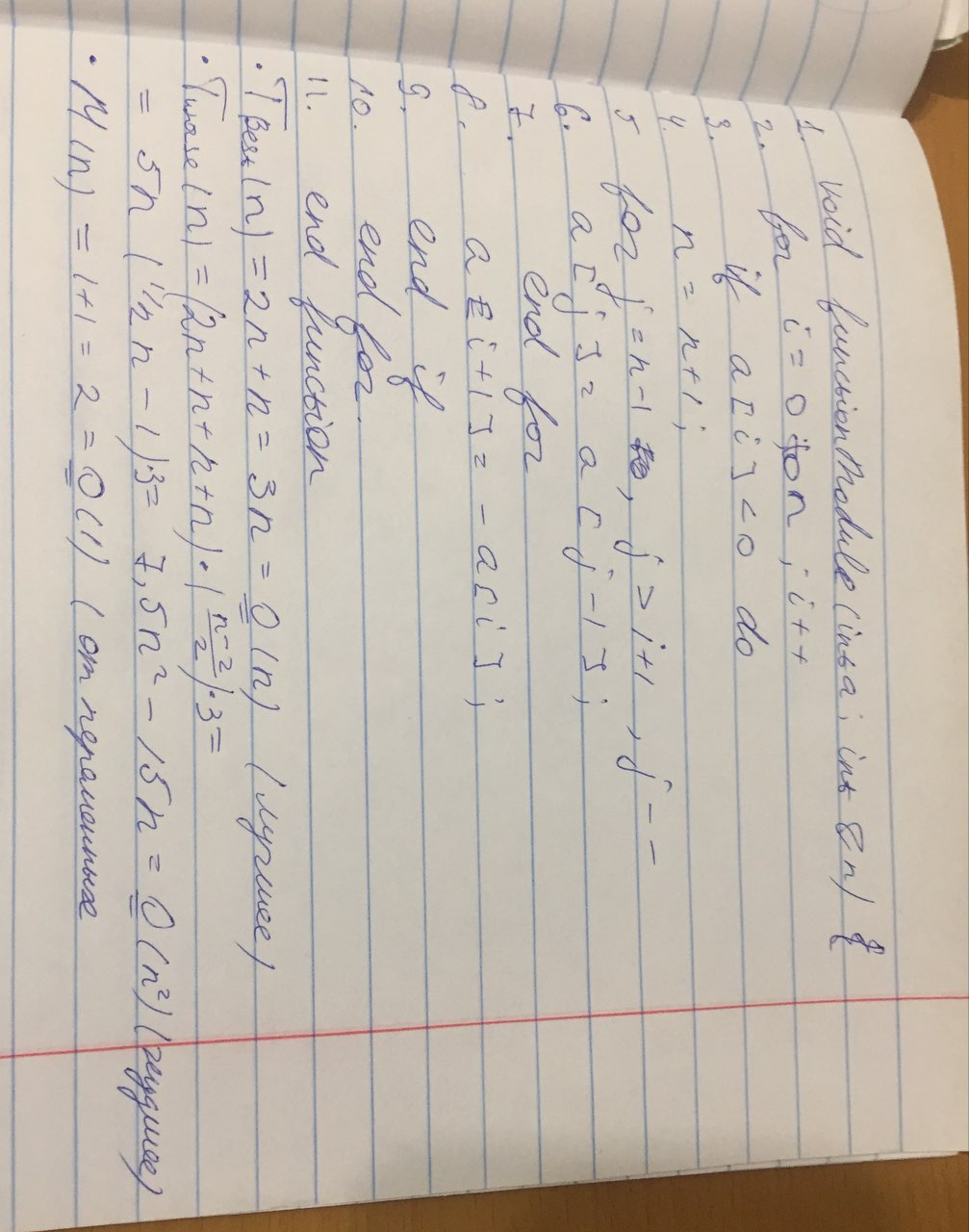
1. Буде -2
2. -9 9 4 -8 8 1 7 -9 9

Так, операції виконуються вірно.

  
Тепер спробуємо перевірити коректність вводу. Для n(від 1 до 100) працює, якщо нижня межа більше верхньої, то користувачу пропонується ввести ще раз та якщо ввести в меню щось інше від 1,2 чи 3 видається помилка.

**Аналіз Алгоритму :**

1)

2.